

```

FFF FFFF FFFF FFFF FFFF      111      111      XXX      XXX
FFF FFFF FFFF FFFF FFFF      111      111      XXX      XXX
FFF FFFF FFFF FFFF FFFF      111      111      XXX      XXX
FFF      111111      111111      XXX      XXX
FFF      111111      111111      XXX      XXX
FFF      111111      111111      XXX      XXX
FFF      111      111      XXX      XXX
FFF      111      111      XXX      XXX
FFF      111      111      XXX      XXX
FFFFFFFF FFFF      111      111      XXX      XXX
FFFFFFFF FFFF      111      111      XXX      XXX
FFFFFFFF FFFF      111      111      XXX      XXX
FFF      111      111      XXX      XXX
FFF      111      111      XXX      XXX
FFF      111      111      XXX      XXX
FFF      111      111      XXX      XXX
FFF      111      111      XXX      XXX
FFF      111      111      XXX      XXX
FFF      111      111      XXX      XXX
FFF      111111111      111111111      XXX      XXX
FFF      111111111      111111111      XXX      XXX
FFF      111111111      111111111      XXX      XXX

```

```
EEEEEEEEEE NN NN TTTTTTTTTT EEEEEEEEEE RRRRRRRR
EEEEEEEEEE NN NN TTTTTTTTTT EEEEEEEEEE RRRRRRRR
EE NN NN NN NN TT TT RR RR
EE NN NN NN NN TT TT RR RR
EE NNNN NN NN TT TT RR RR
EE NNNN NN NN TT TT RR RR
EEEEEEEE NN NN NN TT TT RR RR
EEEEEEEE NN NN NN TT TT RR RR
EE NN NNNN TT TT RR RR
EE NN NNNN TT TT RR RR
EE NN NN TT TT RR RR
EEEEEEEE NN NN TT TT RR RR
EEEEEEEE NN NN TT TT RR RR
```

```
LL LL IIIIII SSSSSSSS
LL LL IIIIII SSSSSSSS
LL LL II SS
LL LL II SS
LL LL II SS
LL LL II SS
LL LL II SSSSSS
LL LL II SSSSSS
LL LL II SS
LL LL II SS
LL LL II SS
LL LL II SS
LLLLLLLLLL IIIIII SSSSSSSS
LLLLLLLLLL IIIIII SSSSSSSS
```

ENTER

M 16
16-Sep-1984 00:21:45
14-Sep-1984 12:30:19

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[F11X.SRC]ENTER.B32;2 Page 1
(1)

```
0001 0 MODULE ENTER (
0002 0     LANGUAGE (BLISS32),
0003 0     IDENT = 'V04-001'
0004 0 ) =
0005 1 BEGIN
0006 1
0007 1
0008 1 *****
0009 1 *
0010 1 *  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0011 1 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0012 1 *  ALL RIGHTS RESERVED.
0013 1 *
0014 1 *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0015 1 *  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0016 1 *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0017 1 *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0018 1 *  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0019 1 *  TRANSFERRED.
0020 1 *
0021 1 *  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0022 1 *  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0023 1 *  CORPORATION.
0024 1 *
0025 1 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0026 1 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0027 1 *
0028 1 *****
0029 1
0030 1
0031 1 ++
0032 1
0033 1 FACILITY: F11ACP Structure Level 2
0034 1
0035 1 ABSTRACT:
0036 1
0037 1     This routine creates a new directory entry for the given file.
0038 1
0039 1 ENVIRONMENT:
0040 1
0041 1     STARLET operating system, including privileged system services
0042 1     and internal exec routines.
0043 1
0044 1 --
0045 1
0046 1
0047 1 AUTHOR: Andrew C. Goldstein, CREATION DATE: 11-Jan-1978 16:50
0048 1
0049 1 MODIFIED BY:
0050 1
0051 1     V04-001 CWH4001 CW Hobbs 9-Aug-1984
0052 1     Fix truncation error.
0053 1
0054 1     V03-006 CDS0003 Christian D. Saether 3-Aug-1984
0055 1     Modify handling of directory index updating.
0056 1
0057 1     V03-005 LMP0252 L. Mark Pilant, 25-Jun-1984 10:03
```


ENTER
V04-001

B 1

16-Sep-1984 00:21:45
14-Sep-1984 12:30:19

VAX-11 Bliss-32 V4.0-742

DISK\$VMSMASTER:[F11X.SRC]ENTER.B32;2 Page 2
(1)

ENT
V04-

```

58      0058 1 1
59      0059 1
60      0060 1
61      0061 1
62      0062 1
63      0063 1
64      0064 1
65      0065 1
66      0066 1
67      0067 1
68      0068 1
69      0069 1
70      0070 1
71      0071 1
72      0072 1
73      0073 1
74      0074 1
75      0075 1
76      0076 1
77      0077 1
78      0078 1
79      0079 1
80      0080 1
81      0081 1
82      0082 1
83      0083 1
84      0084 1
85      0085 1
86      0086 1
87      0087 1
88      0088 1
89      0089 1
90      0090 1
91      0091 1
92      0092 1
93      0093 1
94      0094 1
95      0095 1
96      0096 1
97      0097 1
98      1088 1
99      1089 1
100     1090 1
101     1091 1
102     1092 1
103     1093 1

Make sure that the test for removing a directory entry on an
an auto-purge uses the relative position of the entry, no the
address. This corrects a problem with corrupted directories
for large directories with a version limit of 1.

V03-004 ACG0419      Andrew C. Goldstein,      19-Apr-1984  15:27
Add NOSAFE switch to MAKE_ENTRY to circumvent CSE bug

V03-003 ACG0408      Andrew C. Goldstein,      20-Mar-1984  17:39
Make APPLY_RVN and DEFAULT_RVN macros

V03-002 CDS0002      Christian D. Saether      18-Jan-1984
Modify interface to DEFAULT_RVN and APPLY_RVN.

V03-001 CDS0001      Christian D. Saether      19-Dec-1983
Use BIND_COMMON macro to reduce number of
external COMMON declarations.

V02-012 ACG0264      Andrew C. Goldstein,      12-Feb-1982  15:21
Allow negative version numbers (and treat like zero)

V02-011 ACG0259      Andrew C. Goldstein,      26-Jan-1982  19:17
Fix autopurge when version limit is 1

V02-010 ACG0238      Andrew C. Goldstein,      10-Dec-1981  14:22
Invalidate RMS directory cache when dir is superseded

V02-009 ACG0208      Andrew C. Goldstein,      28-Oct-1981  20:44
Add segmented directory record support

V02-008 ACG34341     Andrew C. Goldstein,      3-Mar-1981   17:06
Fix remove cleanup when first block has been squished

V02-007 ACG0167      Andrew C. Goldstein,      16-Apr-1980  19:26
Previous revision history moved to F11B.REV

**

LIBRARY 'SYSS$LIBRARY:LIB.L32';
REQUIRE 'SRC$:FCPDEF.B32';

FORWARD ROUTINE
ENTER      : L_NORM NOVALUE, : main ENTER routine
MAKE_ENTRY : L_NORM NOVALUE, : build new directory entry
RESTORE_DIR : L_NORM NOVALUE, : restore directory context
```

ENTER
V04-001

C 1
16-Sep-1984 00:21:45
14-Sep-1984 12:30:19

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[F11X.SRC]ENTER.B32;2 Page 3 (2)

```
105 1094 1 GLOBAL ROUTINE ENTER (ABD, FIB, RESULT_LENGTH, RESULT) : L_NORM NOVALUE =
106 1095 1
107 1096 1 ++
108 1097 1
109 1098 1 FUNCTIONAL DESCRIPTION:
110 1099 1
111 1100 1 This routine enters the given file name in the specified directory.
112 1101 1
113 1102 1 CALLING SEQUENCE:
114 1103 1 ENTER (ARG1, ARG2, ARG3, ARG4)
115 1104 1
116 1105 1 INPUT PARAMETERS:
117 1106 1 ARG1: address of buffer descriptor packet
118 1107 1 ARG2: address of FIB of operation
119 1108 1
120 1109 1 IMPLICIT INPUTS:
121 1110 1 NONE
122 1111 1
123 1112 1 OUTPUT PARAMETERS:
124 1113 1 ARG3: address of longword to receive length of result string
125 1114 1 ARG4: address of result string buffer
126 1115 1
127 1116 1 IMPLICIT OUTPUTS:
128 1117 1 DIR_BUFFER: buffer address of current directory block
129 1118 1 DIR_ENTRY: address of directory record
130 1119 1 DIR_VERSION: address of directory version entry
131 1120 1 DIR_END: end of directory data
132 1121 1 PREV_VERSION: version number of superseded entry
133 1122 1
134 1123 1 ROUTINE VALUE:
135 1124 1 NONE
136 1125 1
137 1126 1 SIDE EFFECTS:
138 1127 1 directory altered, result string and length written into buffer packet
139 1128 1
140 1129 1 --
141 1130 1
142 1131 2 BEGIN
143 1132 2
144 1133 2 MAP
145 1134 2 ABD : REF BBLOCKVECTOR [,ABD$C_LENGTH],
146 1135 2 : descriptor list arg
147 1136 2 FIB : REF BBLOCK, : FIB argument
148 1137 2 RESULT : REF VECTOR [,BYTE]; : result string arg
149 1138 2
150 1139 2 LOCAL
151 1140 2 NAME_DESC : BBLOCK [FND_LENGTH], : file name descriptor block
152 1141 2 NAME_BUFFER : VECTOR [FILENAME_LENGTH, BYTE], : buffer for file name
153 1142 2 STATUS, : result of directory search
154 1143 2 P, : string pointer
155 1144 2 VBN, : current VBN of directory file
156 1145 2 NAME_LENGTH; : length of file name, rounded even
157 1146 2
158 1147 2 BIND_COMMON;
159 1148 2
160 1149 2 DIR_CONTEXT_DEF;
161 1150 2
```



```

1151 2 EXTERNAL ROUTINE
1152      PMS_START_SUB : L_NORM,      | start subfunction metering
1153      PMS_END_SUB   : L_NORM,      | end subfunction metering
1154      DIR_ACCESS    : L_NORM,      | access the directory
1155      PARSE_NAME     : L_NORM,      | parse file string
1156      DIR_SCAN       : L_NORM,      | search the directory
1157      UPDATE_DIRSEQ  : L_NORM,      | update directory sequence count in UCB
1158      ADDRESSING_MODE (LONG_RELATIVE),
1159      NEXT_DIR_REC   : L_NORM,      | advance to next matching record
1160      RETURN_DIR     : L_NORM,      | return data to buffer packet
1161      MARK_DIRTY     : L_NORM;      | mark buffer for rewrite
1162
1163
1164      ! Start metering for this subfunction.
1165
1166      PMS_START_SUB (PMS_ENTER);
1167
1168      ! The file ID to be entered must be non-zero.
1169
1170
1171      IF .FIB[FIB$W_FID_NUM] EQL 0
1172      THEN ERR_EXIT (SS$BADPARAM);
1173
1174      ! Find the name string in the buffer packet. Parse the string into the
1175      ! name descriptor block. Mask out the wild card bits, since they are
1176      ! not permitted. Check the version number; it must be positive.
1177
1178
1179      PARSE_NAME (NAME_DESC, NAME_BUFFER, .ABD[ABD$C_NAME, ABD$W_COUNT],
1180      .ABD[ABD$C_NAME, ABD$W_TEXT] + ABD[ABD$C_NAME, ABD$W_TEXT] + 1,
1181      .FIB[FIB$W_NMCTL] AND FIB$M_NEWVER);
1182
1183      IF .NAME_DESC[FND_WILD]
1184      THEN ERR_EXIT (SS$BADFILENAME);
1185
1186      IF .NAME_DESC[FND_VERSION] LSS 0
1187      THEN NAME_DESC[FND_VERSION] = 0;
1188
1189      NAME_LENGTH = .NAME_DESC[FND_COUNT] + 1 AND NOT 1;
1190
1191      ! Access the directory.
1192
1193
1194      DIR_ACCESS (.FIB, 1);
1195
1196      ! Search the directory for the indicated name. If the search succeeds,
1197      ! we have a duplicate entry. If supersede is enabled, do it; otherwise,
1198      ! take an error exit. If the search failed, make a new entry.
1199
1200
1201      DIR_RECORD = 0;
1202      LAST_ENTRY[0] = 0;
1203      STATUS = DIR_SCAN (NAME_DESC, 0, 0, 0, 0, 0, -1);
1204      IF .DIR_VERSION NEQ 0
1205      THEN FIB[FIB$W_VERLIMIT] = .VERSION_LIMIT;
1206
1207      IF .STATUS
1208      AND NOT .NAME_DESC[FND_MAX_VER]
1209      AND .NAME_DESC[FND_VERSION] NEQ 0

```

```
219 1208 2 THEN
220 1209 BEGIN
221 1210 IF NOT .FIB[FIB$V_SUPERSEDE]
222 1211 THEN ERR_EXIT (SS$_DUPFILENAME);
223 1212
224 1213 DIR_END = .DIR_VERSION;
225 1214 PREV_VERSION = .DIR_VERSION[DIR$W_VERSION];
226 1215 CH$MOVE (FIB$S_FID, DIR_VERSION[DIR$W_FID], SUPER_FID);
227 1216 APPLY_RVN (SUPER_FID[FIB$W_RVN], CURRENT_RVN);
228 1217 CH$MOVE (FIB$S_FID, FIB[FIB$W_FID], DIR_VERSION[DIR$W_FID]);
229 1218 DEFAULT_RVN (DIR_VERSION[DIR$W_FID_RVN], CURRENT_RVN);
230 1219 USER_STATUS[0] = SS$_SUPERSEDE;
231 1220 CLEANUP_FLAGS[CLF_SUPERSEDE] = 1;
232 1221
233 1222 ! Determine if the entry being superseded is of the form xxx.DIR;1. If
234 1223 so, bump the directory sequence count in the UCB to invalidate RMS caches.
235 1224
236 1225 IF .DIR_VERSION[DIR$W_VERSION] EQL 1 ! simple tests first
237 1226 THEN
238 1227 BEGIN
239 1228 P = CH$FIND_SUB (.DIR_ENTRY[DIR$B_NAMECOUNT], DIR_ENTRY[DIR$T_NAME],
240 1229 4, UPLIT_BYTE ('.DIR'));
241 1230 IF NOT CH$FAIL (.P)
242 1231 AND .P + 4 EQL DIR_ENTRY[DIR$T_NAME] + .DIR_ENTRY[DIR$B_NAMECOUNT]
243 1232 THEN KERNEL_CALL (UPDATE_DIRSEQ);
244 1233 END;
245 1234 END
246 1235
247 1236 ! The operation is not a supersede. Create the new directory entry.
248 1237
249 1238 ELSE
250 1239 MAKE_ENTRY (NAME_DESC, .FIB);
251 1240
252 1241 ! Determine whether higher or lower versions exist of the new entry.
253 1242 ! This is deduced from the relative position of the new version in the record.
254 1243
255 1244 IF .VERSION_COUNT GTRU 0
256 1245 THEN FIB[FIB$V_HIGHVER] = 1;
257 1246
258 1247 IF .DIR_VERSION LSSA .DIR_ENTRY + .DIR_ENTRY[DIR$W_SIZE] + 2 - DIR$C_VERSION
259 1248 OR
260 1249 BEGIN
261 1250 VBN = .DIR_VBN;
262 1251 NEXT_DIR_REC (.DIR_ENTRY, VBN) NEQ 0
263 1252 END
264 1253 THEN FIB[FIB$V_LOWVER] = 1;
265 1254
266 1255 FIB[FIB$W_VERLIMIT] = .VERSION_LIMIT;
267 1256
268 1257 ! Write out the modified directory block and
269 1258 ! return the resultant directory string to the caller.
270 1259
271 1260
272 1261
273 1262
274 1263
275 1264 KERNEL_CALL (RETURN_DIR, .RESULT_LENGTH, .RESULT, .ABD);
```



```

: 276      1265      2
: 277      1266      2 MARK_DIRTY (.DIR_BUFFER);
: 278      1267      2
: 279      1268      2 PMS_END_SUB ();
: 280      1269      2
: 281      1270      1 END;

```

```
! end of routine ENTER
```

```
.TITLE  ENTER
.IDENT  \V04-001\

.PSECT  $CODE$,NOWRT,2
```

52 49 44 2E 00000 P.AAA: .ASCII \.DIR\

```
.EXTRN PMS_START SUB, PMS_END SUB
.EXTRN DIR_ACCESS, PARSE_NAME
.EXTRN DIR_SCAN, UPDATE_DIRSEQ
.EXTRN NEXT_DIR_REC, RETURN_DIR
.EXTRN MARK_DIRTY
```

```
.ENTRY ENTER, Save R2,R3,R4,R5,R6,R7,R8
```

ENTER	ENTER, SAVE R2, R3, R4, R5, R6, R7, R8	1174
MOVAB	-100(SP), SP	1175
MOVAB	220(BASE), R6	1176
MOVAB	510(BASE), R7	1177
MOVAB	12(R6), R8	1178
PUSHL	#7	1179
CALLS	#1, PMS_START_SUB	1180
MOVL	FIB, R0	1181
TSTW	4(R0)	1182
BNEQ	1\$	1183
CHMU	#20	1184
RET		1185
MOVL	FIB, R0	1186
MOVZWL	20(R0), R1	1187
BICL3	#-513, R1, -(SP)	1188
MOVL	ABD, R1	1189
MOVAB	16(R1), R2	1190
MOVZWL	(R2), R0	1191
PUSHAB	1(R2)[R0]	1192
MOVZWL	18(R1), -(SP)	1193
PUSHAB	NAME_BUFFER	1194
PUSHAB	NAME_DESC	1195
CALLS	#5, PARSE_NAME	1196
BLBC	NAME_DESC+1, 2\$	1197
CHMU	#2072	1198
RET		1199
TSTW	NAME_DESC+12	1200
BGEQ	3\$	1201
CLRW	NAME_DESC+12	1202
ADDL3	#1, NAME_DESC+4, R0	1203
BICB2	#1, NAME_LENGTH	1204
PUSHL	#1	1205
PUSHL	FIB	1206
CALLS	#2, DIR_ACCESS	1207
CLRL	216(BASE)	1208
CLRB	28(R6)	1209

					01FC	00000	
		5E	9C	AE	9E	00002	
		56	00DC	CA	9E	00006	
		57	01FE	CA	9E	0000B	
		58	OC	A6	9E	00010	
				07	DD	00014	
	0000G	CF		01	FB	00016	
		50	08	AC	D0	0001B	
			04	A0	B5	0001F	
				03	12	00022	
				14	BF	00024	
					04	00026	
		50	08	AC	D0	00027	1\$:
		51	14	A0	3C	0002B	
7E		51	FFFFFDFF	8F	CB	0002F	
		51	04	AC	D0	00037	
		52	10	A1	9E	0003B	
		50		62	3C	0003F	
			01	A240	9F	00042	
		7E	12	A1	3C	00046	
			10	AE	9F	0004A	
			64	AE	9F	0004D	
	0000G	CF		05	FB	00050	
		05	55	AE	E9	00055	
			0818	8F	BF	00059	
					04	0005D	
			60	AE	B5	0005E	2\$:
				03	18	00061	
			60	AE	B4	00063	
50		58	AE	01	C1	00066	3\$:
		50		01	8A	0006B	
				01	DD	0006E	
			08	AC	DD	00070	
	0000G	CF		02	FB	00073	
			00D8	CA	D4	00078	
			1C	A6	94	0007C	

ENTER
V04-001

G 1
16-Sep-1984 00:21:45
14-Sep-1984 12:30:19

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[F11X.SRC]ENTER.B32;2

Page 7
(2)

		7E		01	CE	0007F	MNEGL	#1, -(SP)	1201
				7E	7C	00082	CLRQ	-(SP)	
				7E	7C	00084	CLRQ	-(SP)	
				7E	D4	00086	CLRL	-(SP)	
			6C	AE	9F	00088	PUSHAB	NAME_DESC	
	0000G	CF		07	FB	0008B	CALLS	#7, DIR_SCAN	
				68	D5	00090	TSTL	(R8)	1202
				09	13	00092	BEQL	4\$	
		51	08	AC	D0	00094	MOVL	FIB, R1	1203
	2C	A1	18	A6	B0	00098	MOVW	24(R6), 44(R1)	
		03		50	E8	0009D	BLBS	STATUS, 6\$	1205
				00A5	31	000A0	BRW	12\$	
	F8	55		01	E0	000A3	BBS	#1, NAME_DESC+1, 5\$	1206
			60	AE	B5	000A8	TSTW	NAME_DESC+12	1207
				F3	13	000AB	BEQL	5\$	
		50	08	AC	D0	000AD	MOVL	FIB, R0	1210
	05	15		02	E0	000B1	BBS	#2, 21(R0), 7\$	
			0868	8F	BF	000B6	CHMU	#2152	1211
				04	00	000BA	RET		
		10		68	D0	000BB	MOVL	(R8), 16(R6)	1213
	0152	CA	00	B8	32	000BF	CVTWL	20(R8), 338(BASE)	1214
		50		68	D0	000C5	MOVL	(R8), R0	1215
	67	02		06	28	000C8	MOVW	#6, 2(R0), (R7)	
			04	A7	95	000CD	TSTB	4(R7)	1216
				05	12	000D0	BNEQ	8\$	
		04		A0	AA	90	MOVB	-96(BASE), 4(R7)	
		A7	04	A7	91	000D7	CMPB	4(R7), #1	
		01		08	12	000DB	BNEQ	9\$	
				A0	AA	D5	TSTL	-96(BASE)	
				03	12	000E0	BNEQ	9\$	
			04	A7	94	000E2	CLRB	4(R7)	
		51	08	AC	D0	000E5	MOVL	FIB, R1	1217
		50		68	D0	000E9	MOVL	(R8), R0	
	02	A0	04	06	28	000EC	MOVW	#6, 4(R1), 2(R0)	
				68	D0	000F2	MOVL	(R8), R0	1218
A0	AA	06		00	ED	000F5	CMPZV	#0, #8, 6(R0), -96(BASE)	
				03	12	000FC	BNEQ	10\$	
			06	A0	94	000FE	CLRB	6(R0)	
		80		0631	8F	3C	MOVZWL	#1585, -128(BASE)	1219
				20	88	00107	BISB2	#32, (BASE)	1220
		01	00	B8	B1	0010A	CMPW	20(R8), #1	1226
				43	12	0010E	BNEQ	13\$	
		50	08	A6	D0	00110	MOVL	8(R6), R0	1229
		51	05	A0	9A	00114	MOVZBL	5(R0), R1	
06	A0			04	39	00118	MATCHC	#4, P.AAA, R1, 6(R0)	1230
		51	FEDF	03	13	00120	BEQL	11\$	
		53		04	D0	00122	MOVL	#4, R3	
		53		04	C2	00125	SUBL2	#4, R3	
				29	13	00128	BEQL	13\$	1231
		53		04	C0	0012A	ADDL2	#4, R3	1232
		51	08	A6	D0	0012D	MOVL	8(R6), R1	
		50	05	A1	9A	00131	MOVZBL	5(R1), R0	
		50	06	A140	9E	00135	MOVAB	6(R1)(R0), R0	
		50		53	D1	0013A	CMP	R3, R0	
				14	12	0013D	BNEQ	13\$	
00000000G		EF		00	FB	0013F	CALLS	#0, UPDATE_DIRSEQ	1233
				0B	11	00146	BRB	13\$	1205

ENT
V04

ENTER
V04-001

H 1
16-Sep-1984 00:21:45
14-Sep-1984 12:30:19

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[F11X.SRC]ENTER.B32;2

Page 8
(2)

		08	AC	DD	00148	12\$:	PUSHL	FIB	:	1241
		58	AE	9F	0014B		PUSHAB	NAME DESC	:	
0000V	CF		02	FB	0014E		CALLS	#2, MAKE_ENTRY	:	
		1A	A6	B5	00153	13\$:	TSTW	26(R6)	:	1247
			09	13	00156		BEQL	14\$:	
	50	08	AC	D0	00158		MOVL	FIB, R0	:	1248
15	A0	80	8F	88	0015C		BISB2	#128, 21(R0)	:	
	50	08	B6	3C	00161	14\$:	MOVZWL	28(R6), R0	:	1250
	50	08	A6	C0	00165		ADDL2	8(R6), R0	:	
	50		06	C2	00169		SUBL2	#6, R0	:	
	50		68	D1	0016C		CMPL	(R8), R0	:	
			11	1F	0016F		BLSSU	15\$:	
	6E		66	D0	00171		MOVL	(R6), VBN	:	1253
			5E	DD	00174		PUSHL	SP	:	1254
		08	A6	DD	00176		PUSHL	8(R6)	:	
0000G	CF		02	FB	00179		CALLS	#2, NEXT_DIR_REC	:	
			50	D5	0017E		TSTL	R0	:	
			09	13	00180		BEQL	16\$:	
	50	08	AC	D0	00182	15\$:	MOVL	FIB, R0	:	1256
15	A0	40	8F	88	00186		BISB2	#64, 21(R0)	:	
	50	08	AC	D0	0018B	16\$:	MOVL	FIB, R0	:	1258
2C	A0	18	A6	B0	0018F		MOVW	24(R6), 44(R0)	:	
		04	AC	DD	00194		PUSHL	ABD	:	1264
	7E	0C	AC	7D	00197		MOVQ	RESULT_LENGTH, -(SP)	:	
0000G	CF		03	FB	0019B		CALLS	#3, RETURN_DIR	:	
		04	A6	DD	001A0		PUSHL	4(R6)	:	1266
0000G	CF		01	FB	001A3		CALLS	#1, MARK_DIRTY	:	
0000G	CF		00	FB	001A8		CALLS	#0, PMS_END_SUB	:	1268
			04	001AD			RET		:	1270

; Routine Size: 430 bytes. Routine Base: \$CGDE\$ + 0004


```

1271 1 GLOBAL ROUTINE MAKE_ENTRY (NAME_DESC, FIB) : L_NORM NOVALUE =
1272 1
1273 1 ++
1274 1
1275 1 FUNCTIONAL DESCRIPTION:
1276 1
1277 1     This routine makes a new directory entry, either for a new file
1278 1     name or for a new version of an existing file.
1279 1
1280 1
1281 1 CALLING SEQUENCE:
1282 1     MAKE_ENTRY (ARG1, ARG2)
1283 1
1284 1 INPUT PARAMETERS:
1285 1     ARG1: address of file name descriptor block
1286 1     ARG2: address of user FIB
1287 1
1288 1 IMPLICIT INPUTS:
1289 1     DIR_VBN: VBN of current directory block
1290 1     DIR_BUFFER: buffer address of current directory block
1291 1     DIR_ENTRY: address of directory record
1292 1     DIR_VERSION: address of directory version entry
1293 1
1294 1 OUTPUT PARAMETERS:
1295 1     NONE
1296 1
1297 1 IMPLICIT OUTPUTS:
1298 1     DIR_END: end of directory data
1299 1     PREV_VERSION: version number of superseded entry
1300 1
1301 1 ROUTINE VALUE:
1302 1     NONE
1303 1
1304 1 SIDE EFFECTS:
1305 1     directory buffer altered, directory may be extended
1306 1
1307 1 --
1308 1
1309 2 BEGIN
1310 2
1311 2 *****
1312 2 SWITCHES NOSAFE:
1313 2     To prevent incorrect formation of a value CSE on DIR_VERSION across the
1314 2     call to DIR_SCAN in the auto-purge logic. Compiler: T4.0-741.
1315 2 *****
1316 2
1317 2 MAP
1318 2     NAME_DESC      : REF BBLOCK,    ! file name descriptor block
1319 2     FIB            : REF BBLOCK;    ! user FIB
1320 2
1321 2 LOCAL
1322 2     STATUS,        ! misc status value
1323 2     EOF,           ! entering at end of file flag
1324 2     DN             : REF BBLOCK,    ! local pointer to name descriptor
1325 2     NAME_LENGTH,   ! length of file name, rounded even
1326 2     NEW_SIZE,      ! size in bytes of new record
1327 2     VERSIONS,      ! version limit for new directory record

```

```
1328      SAVE_VERSION, ! saved version number of new entry
1329      SAVE_DIR_CONTEXT : BBLOCK [DCX_LENGTH]; ! saved context of enter
1330
1331      BIND_COMMON;
1332
1333      DIR_CONTEXT_DEF;
1334
1335      EXTERNAL ROUTINE
1336          DIR_SCAN      : L_NORM,      ! scan directory file
1337          READ_BLOCK    : L_NORM,      ! read a disk block
1338          REMOVE        : L_NORM,      ! remove a directory entry
1339          SHUFFLE_DIR   : L_NORM,      ! extend the directory
1340          NEXT_REC      : L_NORM,      ! get next directory record
1341          UPDATE_INDXX  : NOVALUE;     ! update directory FCB index
1342
1343      ! Set up the position for the insert. If DIR_ENTRY came back zero, either
1344      ! we scanned off the end of the directory, or this is a cleanup from a REMOVE
1345      ! in which the block DIR_VBN got squished out. We read back the previous
1346      ! block, and set up to append the entry, unless it was the first, in which
1347      ! case we read the first and enter at its beginning. Scan to the end of the
1348      ! records in the block.
1349
1350      EOF = 0;
1351      DN = .NAME_DESC;
1352      NAME_LENGTH = .DN[FND_COUNT] + 1 AND NOT 1;
1353      DIR_END = .DIR_ENTRY;
1354      IF .DIR_ENTRY EQL 0
1355      THEN
1356          BEGIN
1357              IF .DIR_VBN GTRU 1
1358              THEN
1359                  BEGIN
1360                      EOF = .EOF + 1;
1361                      DIR_VBN = .DIR_VBN - 1;
1362                      END;
1363                  DIR_ENTRY = DIR_END = DIR_BUFFER = READ_BLOCK (.DIR_VBN+.DIR_FCB[FCBSL_STLBN]-1,
1364                                                                1, DIRECTORY_TYPE);
1365              END;
1366
1367      UNTIL .DIR_END[DIR$W_SIZE] EQL 65535
1368      DO DIR_END = NEXT_REC (.DIR_END);
1369      IF .EOF
1370      THEN DIR_ENTRY = .DIR_END;
1371      DIR_END = .DIR_END + 2;
1372
1373      ! If there was not a name match, we are constructing a whole new record.
1374      ! Compute the record size and see if there is enough space. If not, extend
1375      ! the directory. Then shuffle down the rest of the records and build the
1376      ! new entry. We update the directory index if this is a new last record
1377      ! in the block.
1378
1379      IF .DIR_VERSION EQL 0
1380      THEN
1381          BEGIN
```



```

397 1385 IF .DN[FND_VERSION] EQL 0
398 1386 THEN DN[FND_VERSION] = .DN[FND_VERSION] + 1;
399 1387
400 1388 NEW_SIZE = DIR$C_LENGTH + DIR$C_VERSION + .NAME_LENGTH;
401 1389 IF .NEW_SIZE GTRO .DIR_BUFFER + 512 - .DIR_END
402 1390 THEN SHUFFLE_DIR (1);
403 1391
404 1392 IF .DIR_ENTRY[DIR$W_SIZE] EQL 65535
405 1393 THEN
406 1394     UPDATE_IDX (.DIR_VBN-1, .DN [FND_COUNT], .DN [FND_STRING],
407 1395                .DIR_FCB);
408 1396
409 1397 CH$MOVE (.DIR_END-.DIR_ENTRY, .DIR_ENTRY, .DIR_ENTRY+.NEW_SIZE);
410 1398
411 1399 DIR_ENTRY[DIR$W_SIZE] = .NEW_SIZE - 2;
412 1400 VERSIONS = .FIB[FIB$W_VERLIMIT];
413 1401 IF .VERSIONS EQL 0
414 1402 THEN VERSIONS = .DIR_FCB[FCB$W_VERSIONS];
415 1403 IF .VERSIONS EQL 0
416 1404 THEN VERSIONS = 32767;
417 1405 DIR_ENTRY[DIR$W_VERLIMIT] = .VERSIONS;
418 1406 VERSION_LIMIT = .VERSIONS;
419 1407
420 1408 DIR_ENTRY[DIR$B_FLAGS] = DIR$C_FID;
421 1409 DIR_ENTRY[DIR$B_NAMECOUNT] = .DN[FND_COUNT];
422 1410 CH$COPY (.DN[FND_COUNT], .DN[FND_STRING],
423 1411          0, .NAME_LENGTH, DIR_ENTRY[DIR$T_NAME]);
424 1412
425 1413 DIR_VERSION = .DIR_ENTRY + .NEW_SIZE - DIR$C_VERSION;
426 1414 END
427 1415
428 1416
429 1417 : Otherwise we are adding a new version to an existing entry. If the
430 1418 version limit is less than maximal, check for version overflow. If
431 1419 so, move out the oldest version to be superseded. We do this by
432 1420 saving the current directory position and scanning to the lowest
433 1421 version of the file. If the scan for lowest version comes back with
434 1422 an error, this indicates that we were already at the lowest version;
435 1423 if so, give an error.
436 1424
437 1425
438 1426 ELSE
439 1427 BEGIN
440 1428 IF .DN[FND_VERSION] EQL 0
441 1429 OR (.DN[FND_MAX_VER] AND .DN[FND_VERSION] LEQU .DIR_VERSION[DIR$W_VERSION])
442 1430 THEN DN[FND_VERSION] = .DIR_VERSION[DIR$W_VERSION] + 1;
443 1431 IF .DN[FND_VERSION] LSS 0
444 1432 THEN ERR_EXIT (SS$_BADFILEVER);
445 1433
446 1434 IF .VERSION_LIMIT LSSU 32767
447 1435 THEN
448 1436 BEGIN
449 1437     SAVE_VERSION = .DN[FND_VERSION];
450 1438     DN[FND_FLAGS] = 0;
451 1439     DN[FND_VERSION] = 32768;
452 1440     CH$MOVE (DCX_LENGTH, DIR_CONTEXT, SAVE_DIR_CONTEXT);
453 1441     STATUS = DIR_SCAN (.DN, 0, .DIR_VBN-1, .DIR_ENTRY, .DIR_VERSION, .DIR_PRED, -1);

```

```
454 1442 4 IF .STATUS
455 1443 4 THEN
456 1444 4 BEGIN
457 1445 4 IF .VERSION_COUNT + 1 GEQU .VERSION_LIMIT
458 1446 4 THEN
459 1447 6 BEGIN
460 1448 6 PREV VERSION = .DIR VERSION[DIR$W VERSION];
461 1449 6 CH$MOVE (.DIR_ENTRY[DIR$B_NAMECOUNT]+1,
462 1450 6 DIR_ENTRY[DIR$B_NAMECOUNT], PREV_NAME);
463 1451 6 CH$MOVE (FIB$S FID, DIR VERSION[DIR$W FID], SUPER_FID);
464 1452 6 APPLY RVN (SUPER_FID[FID$W RVN], .CURRENT_RVN);
465 1453 6 USER STATUS[0] = SSS FILEPORGED;
466 1454 6 CLEARUP_FLAGS[CLF SUPERSEDE] = 1;
467 1455 6 REMOVE T.DIR_VBN EQL .SAVE DIR_CONTEXT[DCX_VBN]
468 1456 6 AND .DIR_ENTRY - .DIR_BUFFER EQL
469 1457 6 .SAVE_DIR_CONTEXT[DCX_ENTRY] - .SAVE_DIR_CONTEXT[DCX_BUFFER]);
470 1458 4 END;
471 1459 4 END
472 1460 4 ELSE
473 1461 4 IF .VERSION_COUNT GEQU .VERSION_LIMIT
474 1462 4 THEN ERR_EXIT (SSS_TOOMANYVER);
475 1463 4
```

476 1464 4 ! Now reposition to the point of the enter to make the new entry.

```
477 1465 4
478 1466 4
479 1467 4 DN[FND VERSION] = .SAVE VERSION;
480 1468 4 RESTORE_DIR (SAVE_DIR_CONTEXT);
481 1469 4 END;
482 1470 4
```

483 1471 4 ! Check available space and extend if necessary. If we are out of space
484 1472 4 in the block, the version is going to the front of the record, the
485 1473 4 record is the only one in its block, and the record has a
486 1474 4 predecessor, we backspace to the previous record and try to append
487 1475 4 the entry. This prevents pathological directory fragmentation when
488 1476 4 new versions are inserted in descending order, as might happen during
489 1477 4 a wild card copy. If this gambit fails or is not applicable, we have
490 1478 4 to split the block.

```
491 1479 4
492 1480 4
493 1481 4 IF DIR$C_VERSION GTRU .DIR_BUFFER + 512 - .DIR_END
494 1482 4 THEN
```

```
495 1483 4 BEGIN
496 1484 4 IF .DIR_ENTRY EQL .DIR_BUFFER
497 1485 4 AND .DIR_VBN GTRU 1
498 1486 4 AND .DIR_VERSION EQL .DIR_ENTRY + .NAME_LENGTH + DIR$C_LENGTH
499 1487 4 AND .BBLOCK [NEXT REC (.DIR_ENTRY), DIR$W_SIZE] EQL 65535
500 1488 4 AND CH$EQL (.DIR_ENTRY[DIR$B_NAMECOUNT]+1,
501 1489 4 DIR_ENTRY[DIR$B_NAMECOUNT],
502 1490 4 .DIR_ENTRY[DIR$B_NAMECOUNT]+1,
503 1491 4 LAST_ENTRY)
```

```
504 1492 4 THEN
505 1493 4 BEGIN
506 1494 4 DIR_VBN = .DIR_VBN - 1;
507 1495 4 DIR_BUFFER = DIR_END = READ_BLOCK (.DIR_VBN-1+.DIR_FCB[FCB$L_STLBN], 1, DIRECTORY_TYPE);
508 1496 4 DO
509 1497 4 BEGIN
510 1498 4 DIR_ENTRY = .DIR_END;
```



```

1499      DIR_END = NEXT_REC (.DIR_END);
1500      END
1501      UNTIL .DIR_END[DIR$W SIZE] EQL 65535;
1502      DIR_VERSION = .DIR_END;
1503      DIR_END = .DIR_END + 2;
1504      IF DIR$C VERSION GTRU .DIR_BUFFER + 512 - .DIR_END
1505      THEN SHUFFLE_DIR (1);
1506      END
1507      ELSE
1508      SHUFFLE_DIR (1);
1509      END;
1510
1511      DIR_ENTRY[DIR$W SIZE] = .DIR_ENTRY[DIR$W SIZE] + DIR$C VERSION;
1512      CH$MOVE (.DIR_END-.DIR_VERSION, .DIR_VERSION, .DIR_VERSION+DIR$C_VERSION);
1513      END;
1514
1515      ! Now insert the version number and file ID into the version slot.
1516      !
1517
1518      CLEANUP_FLAGS[CLF_REMOVE] = 1;
1519      DIR_VERSION[DIR$W_VERSION] = .DN[FND VERSION];
1520      CH$MOVE (FIB$S FID, FIB[FIB$W FID], DIR_VERSION[DIR$W FID]);
1521      DEFAULT_RVN (DIR_VERSION[DIR$W_FID_RVN], .CURRENT_RVN);
1522
1523      ! end of routine MAKE_ENTRY

```

				.EXTRN	READ_BLOCK, REMOVE	
				.EXTRN	SHUFFLE_DIR, NEXT_REC	
				.EXTRN	UPDATE_IDX	
				.ENTRY	MAKE_ENTRY, Save R2,R3,R4,R5,R6,R7,R8,R9,-	1271
					R11	
	SE	90	AE 9E 00002	MOVAB	-112(SP), SP	
		00D0	CA 9F 00006	PUSHAB	208(BASE)	1329
	58	00DC	CA 9E 0000A	MOVAB	220(BASE), R8	
		01FE	CA 9F 0000F	PUSHAB	510(BASE)	
		04	A8 9F 00013	PUSHAB	4(R8)	1331
	5B	08	A8 9E 00016	MOVAB	8(R8), R11	
		0C	A8 9F 0001A	PUSHAB	12(R8)	
		10	A8 9F 0001D	PUSHAB	16(R8)	
		52	D4 00020	CLRL	EOF	1352
	59	04	AC D0 00022	MOVL	NAME_DESC, DN	1353
50	04	A9	01 C1 00026	ADDL3	#1, 4(DN), R0	1354
57		50	01 CB 0002B	BICL3	#1, R0, NAME_LENGTH	
	00	BE	6B D0 0002F	MOVL	(R11), 20(SP)	1355
			6B D5 00033	TSTL	(R11)	1356
		29	12 00035	BNEQ	2\$	
	01		68 D1 00037	CMPL	(R8), #1	1359
		04	1B 0003A	BLEQU	1\$	
		52	D6 0003C	INCL	EOF	1362
		68	D7 0003E	DECL	(R8)	1363
		02	DD 00040	PUSHL	#2	1365
		01	DD 00042	PUSHL	#1	
	50	18	BE D0 00044	MOVL	24(SP), R0	
50	68	30	A0 C1 00048	ADDL3	48(R0), (R8), R0	

			FF	A0	9F	0004D	PUSHAB	-1(R0)		
	0000G	CF		03	FB	00050	CALLS	#3, READ_BLOCK		
		C8		50	D0	00055	MOVL	R0, @8(SP)		
		00		50	D0	00059	MOVL	R0, @0(SP)		
		6B		50	D0	0005D	MOVL	R0, (R11)		
	FFFF	8F	00	BE	D0	00060	MOVL	@0(SP), R0	1369	
				60	B1	00064	CMPW	(R0), #65535		
				0E	13	00069	BEQL	3\$		
	0000G	CF	00	BE	DD	0006B	PUSHL	@0(SP)	1370	
		00		01	FB	0006E	CALLS	#1, NEXT_REC		
		BE		50	D0	00073	MOVL	R0, @0(SP)		
		04		E7	11	00077	BRB	2\$		
		6B	00	52	E9	00079	BLBC	EOF, 4\$	1371	
	00	BE		BE	D0	0007C	MOVL	@0(SP), (R11)	1372	
		50	0C	02	C0	00080	ADDL2	#2, @0(SP)	1373	
			04	A9	9E	00084	MOVAB	12(DN), R0	1385	
				BE	D5	00088	TSTL	@4(SP)	1382	
				03	13	0008B	BEQL	5\$		
				0094	31	0008D	BRW	10\$		
				60	B5	00090	TSTW	(R0)	1385	
				02	12	00092	BNEQ	6\$		
				60	B6	00094	INCW	(R0)	1386	
	50	56	0E	A7	9E	00096	MOVAB	14(R7), NEW_SIZE	1388	
		BE	00	BE	C3	0009A	SUBL3	@0(SP), @8(SP), R0	1389	
		50	0200	C0	9E	000A0	MOVAB	512(R0), R0		
		50		56	D1	000A5	CMPL	NEW_SIZE, R0		
				07	1B	000A8	BLEQU	7\$		
	0000G	CF		01	DD	000AA	PUSHL	#1	1390	
	FFFF	8F	00	01	FB	000AC	CALLS	#1, SHUFFLE_DIR		
				BB	B1	000B1	CMPW	@0(R11), #65535	1392	
				10	12	000B7	BNEQ	8\$		
			10	BE	DD	000B9	PUSHL	@16(SP)	1395	
			04	A9	7D	000BC	MOVQ	4(DN), -(SP)	1394	
	7E	6B		01	C3	000C0	SUBL3	#1, (R8), -(SP)		
		CF		04	FB	000C4	CALLS	#4, UPDATE_IND		
	51	BE	0000G	6B	C3	000C9	SUBL3	(R11), @0(SP), R1	1397	
	00	BB	00	51	28	000CE	MOVCS	R1, @0(R11), @0(R11)[NEW_SIZE]		
00	BB46	56		02	A3	000D5	SUBW3	#2, NEW_SIZE, @0(R11)	1399	
00	BB	50	08	AC	D0	000DA	MOVL	FIB, R0	1400	
		51	2C	A0	3C	000DE	MOVZWL	44(R0), VERSIONS		
				0F	12	000E2	BNEQ	9\$	1401	
		50	10	BE	D0	000E4	MOVL	@16(SP), R0	1402	
		51	40	A0	3C	000E8	MOVZWL	64(R0), VERSIONS		
				05	12	000EC	BNEQ	9\$	1403	
		51	7FFF	8F	3C	000EE	MOVZWL	#32767, VERSIONS	1404	
		50		6B	D0	000F3	MOVL	(R11), R0	1405	
	02	A0		51	B0	000F6	MOVW	VERSIONS, 2(R0)		
	18	A8		51	B0	000FA	MOVW	VERSIONS, 24(R8)	1406	
		50		6B	D0	000FE	MOVL	(R11), R0	1408	
			04	A0	94	00101	CLRB	4(R0)		
		50		6B	D0	00104	MOVL	(R11), R0	1409	
	05	A0	04	A9	90	00107	MOVB	4(DN), 5(R0)		
		50		6B	D0	0010C	MOVL	(R11), R0	1411	
57	00	89	04	A9	2C	0010F	MOVCS	4(DN), @8(DN), #0, NAME_LENGTH, 6(R0)		
			06	A0		00116				
	50	6B		56	C1	00118	ADDL3	NEW_SIZE, (R11), R0	1413	
		BE	F8	A0	9E	0011C	MOVAB	-8(R0), @4(SP)		

				01BB	31	00121		BRW	29\$	1382
				60	B5	00124	10\$:	TSTW	(R0)	1428
				0D	13	00126		BEQL	11\$	
11		69		09	E1	00128		BBC	#9, (DN), 12\$	1429
		51		04	BE	0012C		MOVL	@4(SP), R1	
		61		60	B1	00130		CMPW	(R0), (R1)	
				08	1A	00133		BGTRU	12\$	
		51		04	BE	00135	11\$:	MOVL	@4(SP), R1	1430
60		61		01	A1	00139		ADDW3	#1, (R1), (R0)	
				0C	A9	0013D	12\$:	TSTW	12(DN)	1431
				05	18	00140		BGEQ	13\$	
				0820	8F	00142		CHMU	#2080	1432
						04		RET		
		7FFF	8F	18	A8	00147	13\$:	CMPW	24(R8), #32767	1434
				03	1F	0014D		BLSSU	14\$	
				00D7	31	0014F		BRW	23\$	
		56		0C	A9	00152	14\$:	CVTWL	12(DN), SAVE_VERSION	1437
				69	B4	00156		CLRW	(DN)	1438
14	AE	OC	A9	8000	8F	00158		MOVW	#-32768, 12(DN)	1439
			68	0070	8F	0015E		MOVW3	#112, (R8), SAVE_DIR_CONTEXT	1440
			7E		01	CE		MNEGL	#1, -(SP)	1441
				14	A8	DD		PUSHL	20(R8)	
				0C	BE	DD		PUSHL	@12(SP)	
					6B	DD		PUSHL	(R11)	
		7E	68		01	C3		SUBL3	#1, (R8), -(SP)	
					7E	D4		CLRL	-(SP)	
					59	DD		PUSHL	DN	
		0000G	CF		07	FB		CALLS	#7, DIR_SCAN	
			03		50	EB		BLBS	STATUS, -15\$	1442
					008E	31		BRW	21\$	
			50	1A	A8	3C	15\$:	MOVZWL	26(R8), R0	1445
					50	D6		INCL	R0	
50	18	A8	10		00	ED		CMPZV	#0, #16, 24(R8), R0	
					7E	1A		BGTRU	20\$	
			50	04	BE	DD		MOVL	@4(SP), R0	1448
		0152	CA		60	32		CVTWL	(R0), 338(BASE)	
			50		6B	DD		MOVL	(R11), R0	1449
			51	05	A0	9A		MOVZBL	5(R0), R1	
					51	D6		INCL	R1	
0156	CA	05	A0		51	28		MOVW3	R1, 5(R0), 342(BASE)	1450
			50	04	BE	DD		MOVL	@4(SP), R0	1451
OC	BE	02	A0		06	28		MOVW3	#6, 2(R0), @12(SP)	
	50	OC	AE		04	C1		ADDL3	#4, 12(SP), R0	1452
					60	95		TSTB	(R0)	
					09	12		BNEQ	16\$	
		50	OC	AE	04	C1		ADDL3	#4, 12(SP), R0	
			60	A0	AA	90		MOVB	-96(BASE), (R0)	
		50	OC	AE	04	C1	16\$:	ADDL3	#4, 12(SP), R0	
			01		60	91		CMPB	(R0), #1	
					0C	12		BNEQ	17\$	
				A0	AA	D5		TSTL	-96(BASE)	
					07	12		BNEQ	17\$	
		50	OC	AE	04	C1		ADDL3	#4, 12(SP), R0	
					60	94		CLRB	(R0)	
		80	AA	0679	8F	3C	17\$:	MOVZWL	#1657, -128(BASE)	1453
			6A		20	88		BISB2	#32, (BASE)	1454
					51	D4		CLRL	R1	1455

	14	AE		68	D1	001E7	CMPL	(R8), SAVE_DIR_CONTEXT	
				02	12	001EB	BNEQ	18\$	
50		6B	08	51	D6	001ED	INCL	R1	
52	1C	AE	18	BE	C3	001EF	SUBL3	@8(SP), (R11), R0	1456
				AE	C3	001F4	SUBL3	SAVE_DIR_CONTEXT+4, SAVE_DIR_CONTEXT+8, R2	1457
		52		53	D4	001FA	CLRL	R3	
				50	D1	001FC	CMPL	R0, R2	
				02	12	001FF	BNEQ	19\$	
				53	D6	00201	INCL	R3	
7E		54		51	D2	00203	MCOML	R1, R4	1456
		53		54	CB	00206	BICL3	R4, R3, -(SP)	
	0000G	CF		01	FB	0020A	CALLS	#1, REMOVE	
				0C	11	0020F	BRB	22\$	1442
	18	A8	1A	A8	B1	00211	CMPW	26(R8), 24(R8)	1461
				05	1F	00216	BLSSU	22\$	
			0990	8F	BF	00218	CHMU	#2448	1462
					04	0021C	RET		
	0C	A9		56	B0	0021D	MOVW	SAVE_VERSION, 12(DN)	1467
			14	AE	9F	00221	PUSHAB	SAVE_DIR_CONTEXT	1468
	0000V	CF		01	FB	00224	CALLS	#1, RESTORE DIR	
50	08	BE	00	BE	C3	00229	SUBL3	@0(SP), @8(SP), R0	1481
		50	0200	C0	9E	0022F	MOVAB	512(R0), R0	
		08		50	D1	00234	CMPL	R0, #8	
				03	1F	00237	BLSSU	24\$	
				0091	31	00239	BRW	28\$	
	08	BE		6B	D1	0023C	CMPL	(R11), @8(SP)	1484
				2F	12	00240	BNEQ	25\$	
		01		68	D1	00242	CMPL	(R8), #1	1485
				7F	1B	00245	BLEQU	27\$	
50		6B		57	C1	00247	ADDL3	NAME_LENGTH, (R11), R0	1486
		50		06	C0	0024B	ADDL2	#6, R0	
		50	04	BE	D1	0024E	CMPL	@4(SP), R0	
				72	12	00252	BNEQ	27\$	
				6B	DD	00254	PUSHL	(R11)	1487
	0000G	CF		01	FB	00256	CALLS	#1, NEXT_REC	
	FFFF	8F		60	B1	0025B	CMPW	(R0), #65535	
				64	12	00260	BNEQ	27\$	
		50		6B	D0	00262	MOVL	(R11), R0	1488
		51	05	A0	9A	00265	MOVZBL	5(R0), R1	
				51	D6	00269	INCL	R1	
1C	A8	05	A0	51	29	0026B	CMPC3	R1, 5(R0), 28(R8)	1489
				53	12	00271	BNEQ	27\$	
				68	D7	00273	DECL	(R8)	1494
				02	DD	00275	PUSHL	#2	1495
				01	DD	00277	PUSHL	#1	
		50	18	BE	D0	00279	MOVL	@24(SP), R0	
58		6B	30	A0	C1	0027D	ADDL3	48(R0), (R8), R8	
			FF	A8	9F	00282	PUSHAB	-1(R8)	
	0000G	CF		03	FB	00285	CALLS	#3, READ_BLOCK	
	00	BE		50	D0	0028A	MOVL	R0, @0(SP)	
	08	BE		50	D0	0028E	MOVL	R0, @8(SP)	
		6B	00	BE	D0	00292	MOVL	@0(SP), (R11)	1498
			00	BE	DD	00296	PUSHL	@0(SP)	1499
	0000G	CF		01	FB	00299	CALLS	#1, NEXT_REC	
	00	BE		50	D0	0029E	MOVL	R0, @0(SP)	
		50	00	BE	D0	002A2	MOVL	@0(SP), R0	1501
	FFFF	8F		60	B1	002A6	CMPW	(R0), #65535	

ENTER
V04-001

D 2
16-Sep-1984 00:21:45
14-Sep-1984 12:30:19

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[F11X.SRC]ENTER.B32;2

Page 17
(3)

		04	BE	00	E5	12	002AB	BNEQ	26\$		
		00	BE		BE	D0	002AD	MOVL	@0(SP), @4(SP)		1502
		08	BE	00	02	C0	002B2	ADDL2	#2, @0(SP)		1503
	50		50	0200	BE	C3	002B6	SUBL3	@0(SP), @8(SP), R0		1504
			08		C0	9E	002BC	MOVAB	512(R0), R0		
					50	D1	002C1	CMPL	R0, #8		
					07	1E	002C4	BGEQU	28\$		
		0000G	CF		01	DD	002C6	PUSHL	#1		1508
		00	BB		01	FB	002C8	CALLS	#1, SHUFFLE_DIR		
			50		08	A0	002CD	ADDW2	#8, @0(R11)		1511
	51		00	04	BE	D0	002D1	MOVL	@4(SP), R0		1512
08	A0		60		50	C3	002D5	SUBL3	R0, @0(SP), R1		
		02	AA	A0	51	28	002DA	MOVC3	R1, (R0), 8(R0)		
			50		8F	88	002DF	BISB2	#64, 2(BASE)		1518
			60	04	BE	D0	002E4	MOVL	@4(SP), R0		1519
			51	0C	A9	B0	002E8	MOVW	12(DN), (R0)		
			50	08	AC	D0	002EC	MOVL	FIB, R1		1520
02	A0	04	A1	04	BE	D0	002F0	MOVL	@4(SP), R0		
			50		06	28	002F4	MOVC3	#6, 4(R1), 2(R0)		
A0	AA	06	A0	04	BE	D0	002FA	MOVL	@4(SP), R0		1521
			08		00	ED	002FE	CMPZV	#0, #8, 6(R0), -96(BASE)		
				06	03	12	00305	BNEQ	30\$		
					A0	94	00307	CLRB	6(R0)		
					04	0030A	30\$:	RET			1523

; Routine Size: 779 bytes, Routine Base: \$CODE\$ + 01B2


```

537 1524 1 GLOBAL ROUTINE RESTORE_DIR (CONTEXT) : L_NORM NOVALUE =
538 1525 1
539 1526 1 ++
540 1527 1
541 1528 1 FUNCTIONAL DESCRIPTION:
542 1529 1
543 1530 1     This routine repositions a directory to the location specified
544 1531 1     in the context block and copies the context back into the
545 1532 1     main context block.
546 1533 1
547 1534 1 CALLING SEQUENCE:
548 1535 1     RESTORE_DIR (ARG1)
549 1536 1
550 1537 1 INPUT PARAMETERS:
551 1538 1     ARG1: address of context block to use
552 1539 1
553 1540 1 IMPLICIT INPUTS:
554 1541 1     NONE
555 1542 1
556 1543 1 OUTPUT PARAMETERS:
557 1544 1     NONE
558 1545 1
559 1546 1 IMPLICIT OUTPUTS:
560 1547 1     DIR_CONTEXT: receives contents of supplied context
561 1548 1
562 1549 1 ROUTINE VALUE:
563 1550 1     NONE
564 1551 1
565 1552 1 SIDE EFFECTS:
566 1553 1     directory block read
567 1554 1
568 1555 1 --
569 1556 1
570 1557 2 BEGIN
571 1558 2
572 1559 2 MAP
573 1560 2     CONTEXT          : REF BBLOCK;    ! directory context arg
574 1561 2
575 1562 2 BIND_COMMON;
576 1563 2
577 1564 2 DIR_CONTEXT_DEF;
578 1565 2
579 1566 2 EXTERNAL ROUTINE
580 1567 2     READ_BLOCK        : L_NORM;        ! read a disk block
581 1568 2
582 1569 2
583 1570 2 ! We have to reread the block if either the supplied context is the
584 1571 2 ! main context (indicating a required reposition) or the supplied
585 1572 2 ! context points to a different block than the main context (indicating
586 1573 2 ! that we have actually moved).
587 1574 2
588 1575 2
589 1576 2 IF .CONTEXT EQL DIR_CONTEXT
590 1577 2 OR .CONTEXT[DCX_VBN] NEQ .DIR_VBN
591 1578 2 OR .CONTEXT[DCX_BUFFER] NEQ .DIR_BUFFER
592 1579 2 THEN
593 1580 2     BEGIN

```

```

594 1581 3 DIR_VBN = .CONTEXT[DCX_VBN];
595 1582 3 DIR_ENTRY = .CONTEXT[DCX_ENTRY] - .CONTEXT[DCX_BUFFER];
596 1583 3 IF .CONTEXT[DCX_VERSION] NEQ 0
597 1584 3 THEN DIR_VERSION = .CONTEXT[DCX_VERSION] - .CONTEXT[DCX_BUFFER];
598 1585 3 IF .CONTEXT[DCX_END] NEQ 0
599 1586 3 THEN DIR_END = .CONTEXT[DCX_END] - .CONTEXT[DCX_BUFFER];
600 1587 3 IF .CONTEXT[DCX_PRED] NEQ 0
601 1588 3 THEN DIR_PRED = .CONTEXT[DCX_PRED] - .CONTEXT[DCX_BUFFER];
602 1589 3 DIR_BUFFER = READ_BLOCK (.DIR_VBN+.DIR_FCB[FCB$L_STLBN]-1, 1, DIRECTORY_TYPE);
603 1590 3 DIR_ENTRY = .DIR_ENTRY + .DIR_BUFFER;
604 1591 3 IF .DIR_VERSION NEQ 0
605 1592 3 THEN DIR_VERSION = .DIR_VERSION + .DIR_BUFFER;
606 1593 3 IF .DIR_END NEQ 0
607 1594 3 THEN DIR_END = .DIR_END + .DIR_BUFFER;
608 1595 3 IF .DIR_PRED NEQ 0
609 1596 3 THEN DIR_PRED = .DIR_PRED + .DIR_BUFFER;
610 1597 3 VERSION_LIMIT = .CONTEXT[DCX_VERSION_LIMIT];
611 1598 3 VERSION_COUNT = .CONTEXT[DCX_VERSION_COUNT];
612 1599 3 CH$MOVE (FILENAME_LENGTH+1, .CONTEXT[DCX_NAME], LAST_ENTRY);
613 1600 3 END
614 1601 2 ELSE
615 1602 2 CH$MOVE (DCX_LENGTH, .CONTEXT, DIR_CONTEXT);
616 1603 2
617 1604 1 END;

```

! End of routine RESTORE_DIR

				003C 00000	.ENTRY	RESTORE DIR, Save R2,R3,R4,R5		1524
		52	00DC	CA 9E 00002	MOVAB	220(BASE), R2		1560
		53	04	A2 9E 00007	MOVAB	4(R2), R3		1562
		50	04	AC D0 0000B	MOVL	CONTEXT, R0		1576
		52		50 D1 0000F	CMPL	R0, R2		
				0E 13 00012	BEQL	1\$		
		62		60 D1 00014	CMPL	(R0), (R2)		1577
				09 12 00017	BNEQ	1\$		
		63	04	A0 D1 00019	CMPL	4(R0), (R3)		1578
				03 12 0001D	BNEQ	1\$		
				0095 31 0001F	BRW	8\$		
		62		60 D0 00022	MOVL	(R0), (R2)		1581
		50	04	AC D0 00025	MOVL	CONTEXT, R0		1582
08	A2	08	A0	04 A0 C3 00029	SUBL3	4(R0), 8(R0), 8(R2)		
		50	04	AC D0 00030	MOVL	CONTEXT, R0		1583
			0C	A0 D5 00034	TSTL	12(R0)		
				07 13 00037	BEQL	2\$		
0C	A2	0C	A0	04 A0 C3 00039	SUBL3	4(R0), 12(R0), 12(R2)		1584
		50	04	AC D0 00040	MOVL	CONTEXT, R0		1585
			10	A0 D5 00044	TSTL	16(R0)		
				07 13 00047	BEQL	3\$		
10	A2	10	A0	04 A0 C3 00049	SUBL3	4(R0), 16(R0), 16(R2)		1586
		50	04	AC D0 00050	MOVL	CONTEXT, R0		1587
			14	A0 D5 00054	TSTL	20(R0)		
				07 13 00057	BEQL	4\$		
14	A2	14	A0	04 A0 C3 00059	SUBL3	4(R0), 20(R0), 20(R2)		1588
				02 DD 00060	PUSHL	#2		1589
				01 DD 00062	PUSHL	#1		

ENTER
V04-001

G 2
16-Sep-1984 00:21:45
14-Sep-1984 12:30:19

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[F11X.SRC]ENTER.B32;2

Page 20
(4)

50	50	00D0	CA	D0	00064	MOVL	208(BASE), R0	
	62	30	A0	C1	00069	ADDL3	48(R0), (R2), R0	
		FF	A0	9F	0006E	PUSHAB	-1(R0)	
	0000G	CF	03	FB	00071	CALLS	#3, READ_BLOCK	
	63		50	D0	00076	MOVL	R0, (R3)	
	08	A2	63	C0	00079	ADDL2	(R3), 8(R2)	1590
			0C	A2	D5	TSTL	12(R2)	1591
			04	13	00080	BEQL	5\$	
	0C	A2	63	C0	00082	ADDL2	(R3), 12(R2)	1592
			10	A2	D5	TSTL	16(R2)	1593
			04	13	00089	BEQL	6\$	
	10	A2	63	C0	0008B	ADDL2	(R3), 16(R2)	1594
			14	A2	D5	TSTL	20(R2)	1595
			04	13	00092	BEQL	7\$	
	14	A2	63	C0	00094	ADDL2	(R3), 20(R2)	1596
			04	AC	D0	MOVW	CONTEXT, R0	1597
	18	A2	18	A0	B0	MOVW	24(R0), 24(R2)	
			04	AC	D0	MOVW	CONTEXT, R0	1598
	1A	A2	1A	A0	B0	MOVW	26(R0), 26(R2)	
			04	AC	D0	MOVW	CONTEXT, R0	1599
1C	A2	1C	A0	0051	8F	MOVW	#81, 28(R0), 28(R2)	
					04	RET		1576
	62		60	0070	8F	MOVW	#112, (R0), (R2)	1602
					04	RET		1604

; Routine Size: 190 bytes, Routine Base: \$CODE\$ + 04BD

: 618 1605 1
: 619 1606 1 END
: 620 1607 0 ELUDOM

PSECT SUMMARY

Name	Bytes	Attributes
\$CODE\$	1403	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)

Library Statistics

File	Total	Symbols Loaded	Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[SYSLIB]LIB.L32;1	18619	51	0	1000	00:01.9

ENTER
V04-001

H 2
16-Sep-1984 00:21:45
14-Sep-1984 12:30:19

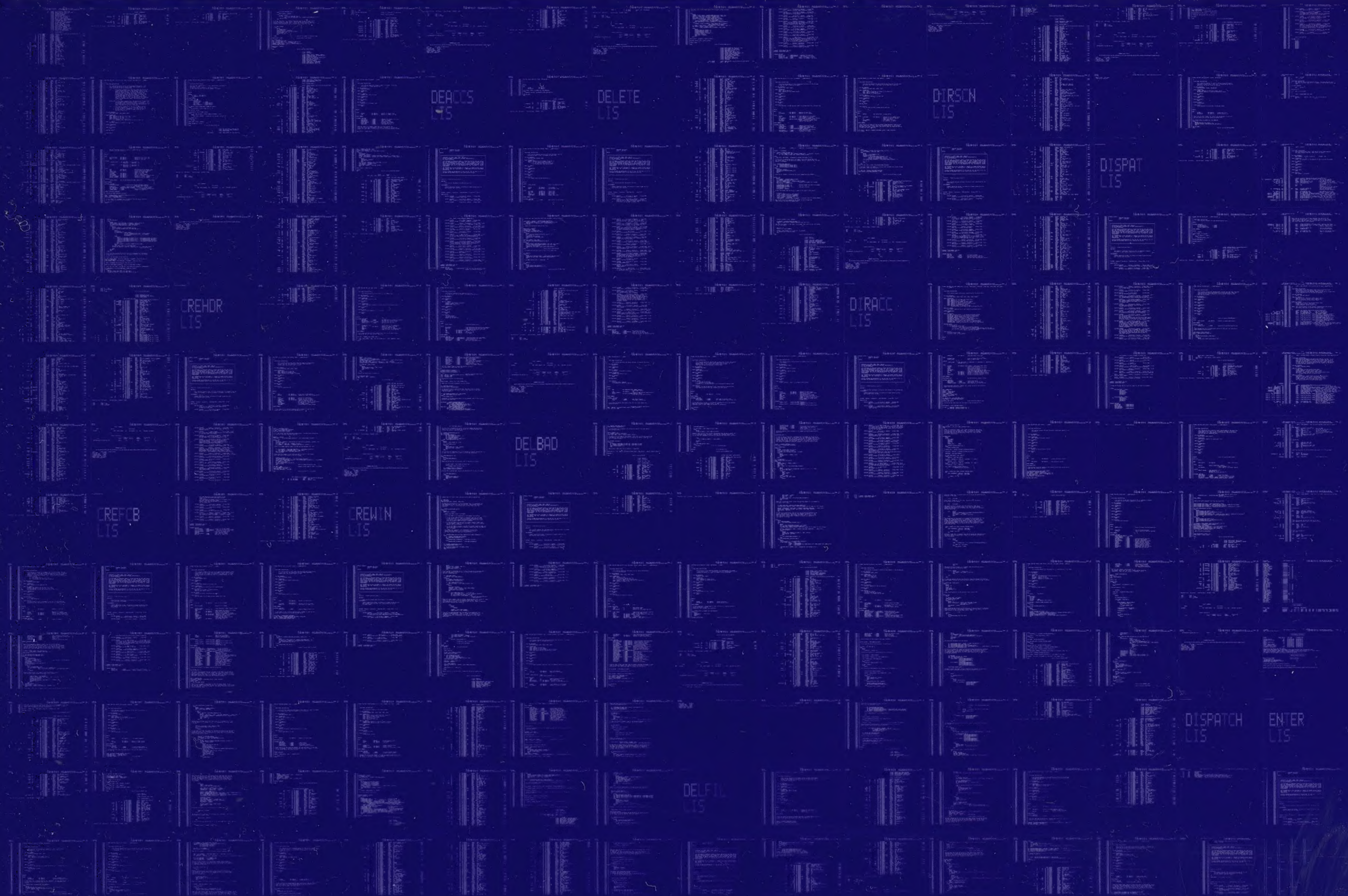
VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[F11X.SRC]ENTER.B32;2 Page 21
(4)

**F

```
:  
:          COMMAND QUALIFIERS  
:  
:  BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LISS:ENTER/OBJ=OBJ$:ENTER MSRC$:ENTER/UPDATE=(ENHS:ENTER)  
:  
: Size:          1399 code + 4 data bytes  
: Run Time:      00:51.8  
: Elapsed Time:  01:35.9  
: Lines/CPU Min: 1861  
: Lexemes/CPU-Min: 47008  
: Memory Used:  377 pages  
: Compilation Complete
```


0169 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY



0170

AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY